An Efficient Variable Length Coding Scheme for an IID Source*

Kar-Ming Cheung Aaron Kiely

kmcheung@shannon.jpl.nasa.gov aaron@shannon.jpl.nasa.gov

Tel: 818-393-9480 Tel: 818-354-6951

Mail Stop 238-420, Jet Propulsion Laboratory, Pasadena, CA 91109

add Calif... I... of T...

Abstract- -- In this article we examine a scheme that uses two alternating Huffman codes to encode a discrete independent and identically distributed source with a dominant symbol, One Huffman code encodes the length of runs of the dominant symbol, the other encodes the remaining symbols. We call this combined strategy alternating runlength Huffman (ARH) coding. This is a popular scheme, used for example in the Efficient Pyramid Image Coder (EPIC) subband coding algorithm.

Since the runlengths of the dominant, symbol are geometrically distributed, they can be encoded using the Huffman codes identified by Golomb [1] and later generalized by Gallager and Van Voorhis [2]. This runlength encoding allows the most likely symbol to be encoded using less than one bit per sample, providing a simple method of overcoming a drawback of prefix codes- that the redundancy approaches one as the largest symbol probability P approaches one. For ARH coding, the redundancy approaches zero as P approaches one.

Comparing the average code rate of ARH with direct Huffman coding we find that

1. If $P < 1/3$, ARH is less efficient than Huffman coding.
2. If $1/3 \leq P < 2/5$, ARH is less efficient than Huffman coding, depending on the source distribution.
3. If $2/5 \leq P \leq 0.618$, ARH and Huffman coding are equally efficient.
4. If $P > 0.618$, ARH is more efficient than Huffman coding,

We give examples of applying ARH coding to some specific sources.

*The research described in this abstract was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract, with the National Aeronautics and Space Administration.

1 Introduction

Consider a discrete source with one symbol that occurs significantly more often than the others. This situation is common in many data compression applications. For example, in predictive coding the error samples tend to be small, or in transform based coding systems, e.g. subband coding or discrete cosine transform, the signal energy is usually more concentrated in the low frequency components. In each of these cases, the data are quantized before compression, and the quantized data is often a low entropy source consisting of long runs of zeros interspersed with small nonzero values.

A well-known drawback of Huffman coding in this case is that while the entropy may be near zero, the rate of a Huffman code can never be less than one since the Huffman code assigns a codeword to every source symbol. A common means of tackling this problem is to encode the runs of the most probable symbol separately from the other symbols. This is more efficient than Huffman coding when one symbol is highly probable.

For example, the sequence 0,1,0,0,0,2, --1 can be thought of as 0^1, 1,03,2,0°, –1, i.e., a sequence that alternates between the separate alphabets $\{0^i : i \geq$ O$\}$ and the nonzero integers. We insert the symbol $()^0$ between adjacent 1101IZWO symbols to ensure that the sequence is alternating. This allows us to encode the runs of zeros and the nonzero values using two different Huffman codes. We call this approach alternating runlength Huffman (ARH) coding. This strategy is particularly suitable for a one-pass adaptive implementation, which we will discuss in a companion paper [3].

Combined runlength and Huffman coding is popular in many well-know]l compression systems. Variations of this technique are used in the baseline *Joint Photographic Expert Group* (JPEG) algorithm [4], the *Efficient Pyramid Image Coder* (EPIC) subband coding algorithm [5], and other applications. The baseline JPEG scheme combines each run of zeros with the subsequent nonzero value and encodes them together using a static Huffman code. The JPEG algorithm provides reasonably good compression performance for most natural images [6]. The EPIC subband coding system encodes the runs of zeros and the non-zero symbols using two separate Huffman codes. Both Huffman code tables are sent with the compressed data [5].

Despite the popularity of combined runlength and Huffman codes, little analysis has been done in this area. The goal of this paper is to quantify the compression efficiency of ARH coding, and to compare it to direct Huffman coding for IID sources. The results presented here are applicable to predictive coding and subband coding systems when the quantized output to the entropy coder is 111). For block transform codings like Discrete Cosine Transform and Karhunen-Loeve Transform, we develop similar results in an upcoming paper [7] by taking into account the fact that the signal energy is varied in the different frequency components.

In Section 2 we evaluate the rate and redundancy of the ARH scheme, and derive a tight bound on ARH redundancy as a function of the largest symbol probability. We compare ARH with direct Huffman coding, and show how the largest symbol probability determines whether ARH or direct Huffman coding is more efficient.

In Section 3, we use the results of Section 2 to find a bound on ARH rate. We also give examples of applying ARH coding to two specific sources.

In Section 4, we discuss future work and give some concluding remarks.

2 Alternating Runlength Huffman Coding

2.1 Rate of ARH coding

Let $\mathcal{A}$ denote a discrete IID source, and let S denote the source symbol with largest probability P. The Huffman coding algorithm applied to $\mathcal{A}$ assigns a codeword to each symbol in a way that minimizes the expected codeword length (or code rate), which we denote by $R_H(\mathcal{A})$. We refer to the application of a Huffman code directly to $\mathcal{A}$ (i.e., without first combining source symbols) as direct Huffman coding.

The redundancy of direct Huffman coding is $\rho_H(\mathcal{A}) = R_H(\mathcal{A}) - \mathcal{H}(\mathcal{A})$, where $\mathcal{H}(\mathcal{A})$ denotes the entropy of $\mathcal{A}$. There are myriad bounds on bounds on $R_H(\mathcal{A})$ and $\rho_H(\mathcal{A})$ [9, 10, 11, 12, 13, 14, 15, 16]. It is well-known that $\rho_H(\mathcal{A})$ approaches 1 as P approaches 1. This is a consequence of the fact that a Huffman code assigns a codeword to each symbol, and hence cannot have rate less than one, no matter how small the entropy.

Let $\mathcal{A}'$ denote the source formed by deleting each occurrence of S from $\mathcal{A}$. Because $\mathcal{A}$ is IID, the runlength i of the most probable symbol S is distributed according to the geometric distribution

$$\mathrm{Prob}[S^i] = (1 - P)P^i.$$

We encode these runlengths using the optimal (Huffman) encoding for this distribution: the Gallager and Van Voorhis (GVH) code [2]. The remaining symbols (i.e., the source $\mathcal{A}'$) is separately encoded using direct Huffman coding.

The GVH codeword for runlength i consists of j zeros followed by a one and a codeword that represents $i \bmod \ell$, where

$$\ell \triangleq \left\lceil \frac{-\log(1+P)}{\log P} \right\rceil$$

and $j \triangleq \lfloor i/\ell \rfloor$. From [2], the average length of a GVH codeword is

$$L(P) = 1 + \lfloor \log_2 \ell \rfloor + \frac{P^k}{1 - P^\ell}$$

where

$$k \triangleq 2^{\lfloor \log_2 \ell \rfloor + 1} - \ell.$$

The particular GVH code can be identified completely by the parameter ℓ. This is especially convenient for adaptive implementations because it eliminates the need to transmit a long code table.

To determine the rate of ARH coding, consider the encoding of the source sequence $\underbrace{SS\cdots S}_{i}X$, where X is some symbol from $\mathcal{A}'$. The expected number of source symbols in such a sequence is

$$E[1+i] = 1 + \sum_{i=0}^{\infty} i(1-P)P^i = \frac{1}{1-P},$$

and the expected number of encoded bits for the sequence is $R_H(\mathcal{A}') + L(P)$. Thus the average rate of ARH is

$$R_{\text{ARH}}(\mathcal{A}) = (1-P)\left[L(P) + R_H(\mathcal{A}')\right] \quad \text{(bits/source sample)}. \tag{1}$$

Bounds on rate (or redundancy) of Huffman codes can be used to bound the quantity $R_H(\mathcal{A}')$ in the above expression, which produces a bound the rate of ARH coding. We give an example of such a bound in Section 3.

2.2 Comparing ARH and direct Huffman Coding

To determine whether ARH or direct Huffman coding is more efficient, we want to compare (1) to $R_H(\mathcal{A})$. If $P \geq 2/5$, a Huffman code for $\mathcal{A}$ assigns a one bit codeword to the most probable symbol [9, Theorem 1], [8, p. 122 Problem 22], producing $R_H(\mathcal{A}) = 1 + (1-P)R_H(\mathcal{A}')$. Comparing this quantity to (1), we find that for this range of P, ARH is more efficient than direct Huffman coding whenever

$$(1-P)L(P) < 1$$

which occurs when

$$P > \gamma \triangleq \frac{\sqrt{5}-1}{2} \approx .618.$$

The quantity $1+\gamma$ is the famous "golden ratio."

When $P \leq \gamma$, we find that $L(P) = 1/(1-P)$ and (1) becomes

$$1 + (1-P)R_H(\mathcal{A}').$$

in fact the ARH strategy in this case reduces to encoding; a zero each time the source emits S, otherwise encoding a one followed by the Huffman codeword for the less probable symbol. Thus, this strategy assigns a binary codeword to each source symbol, and hence can never outperform the optimal assignment, which is obtained by direct Huffman coding.

So if a Huffman code for $\mathcal{A}$ assigns a one bit codeword to S, then it will have the same performance as ARH, otherwise it will perform better. If $2/5 < P < \gamma$, then the ARH strategy is in fact an optimal Huffman code for $\mathcal{A}$. If $P < 1/3$, then the most probable symbol has a codeword of length at least two [8, p. 122 Problem 22], and thus in this case ARH will always be less efficient than direct Huffman coding.

If $1/3 < P < 2/5$, then the Huffman codeword length for the most probable symbol depends on the probability distribution on $\mathcal{A}$, not just on P. For example, applying the Huffman algorithm to the distributions {3/8, 1/5, 1 /5, 1/5, 1/40} and {3/8, 5/32, 5/32, 5/32, 5/32} wc find that the most probable symbol has a codeword of length two bits for the first distribution and only onc bit for the second, even though P = 3/8 in both cases.

'J)o summarize, for any 1111 source with largest symbol probability P,

1. If P <1 /3, ARH is lcss efficient than direct 1 luffman coding.

2. If $1/3 \leq P < 2/5$, the efficiency of ARH is less than or equal to that of direct Huffman coding, depending on the probability distribution] on $\mathcal{A}$.

3. If $2/5 \leq P \leq \gamma$, ARH and direct Huffman coding arc equally efficient.

4. If $P > \gamma$, ARH is more efficient than direct Huffman coding.

Thus in a practical system, wc do not require a great deal of accuracy in determining the the largest source symbol probability to select which coding scheme to usc. It is sufficient to determine whether the most probable symbol has probability less than 2/5 or greater than γ.

2.3 Redundancy of ARH coding

From the grouping property of entropy, $\mathcal{H}(\mathcal{A}) = \mathcal{H}_2(P) + (1 - P)\mathcal{H}(\mathcal{A}')$, where $\mathcal{H}_2$ denotes the binary entropy function. Thus the redundancy of ARH is

$$\begin{aligned}
\rho_{\text{ARH}}(\mathcal{A}) &= R_{\text{ARH}}(\mathcal{A}) - \mathcal{H}(\mathcal{A}) \\
&= (1 - P)[L(P) + R_H(\mathcal{A}')] - \mathcal{H}_2(P) - (1 - P)\mathcal{H}(\mathcal{A}') \\
&= (1 - P)\,[L(P) + \rho_H(\mathcal{A}')] - \mathcal{H}_2(P) \qquad (2) \\
&< (1 - P)[L(P) + 1] - \mathcal{H}_2(P). \qquad (3)
\end{aligned}$$

The inequality follows because the redundancy of a Huffman code is lcss than one. This bound is monotonically decreasing, and not surprising y, shows that the redundancy (and hence the rate) of A RH approaches zero as P approaches one, The bound is tight for any $P > 2/5$: the redundancy of ARH coding for the distribution $\{P, 1 - P - \varepsilon, \varepsilon\}$ approaches (3) as $\varepsilon \to 0$. We show the bound (3) in Figure 1 along with an upper bound on the redundancy of direct Huffman coding [16]. This figure shows that a coding system capable of switching between ARH and direct, Huffman coding depending on the value of P has maximum redundancy of 1/2 occurring at $P = 1/2$.

3 Examples

3.1 Bound on rate in terms of the alphabet size

Suppose the source alphabet has (finite) size n. Then $R_H(\mathcal{A}') \leq \lceil \log_2(n - 1) \rceil$, thus from (1) ,

$$R_{ARH}(\mathcal{A}) \leq (1 - P)[L(P) + \lceil \log_2(n - 1) \rceil] .$$

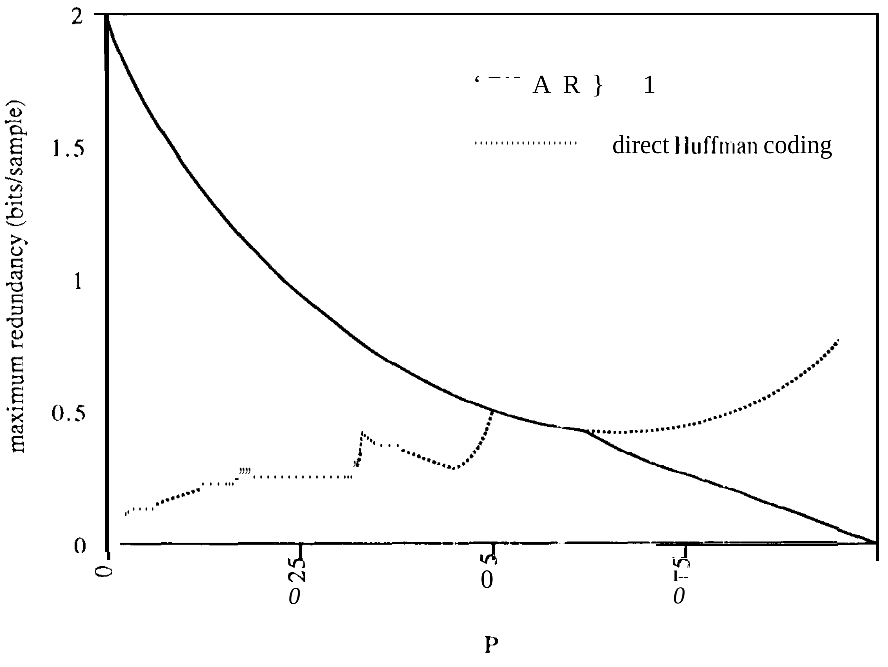

Figure 1: Upper bounds on redundancy of ARH and direct Huffman coding.

Figure 2 shows this upper bound for n = 4, 16, 64, and 256. These upper bounds may be very 100 SC, particularly when $\mathcal{H}(\mathcal{A}')$ is small.

3.2 Geometric source

Suppose that the source alphabet is the set, of non-negative integers, and that the source $\mathcal{A}$ has finite mean. In this case, the geometric distribution 1 'rob [z] $= P(1-P)^i$ maximizes the source entropy [8]. Let $\mathcal{G}_1$ denote this source.

$\mathcal{G}_1$ has entropy $\mathcal{H}(\mathcal{G}_1) = \mathcal{H}_2(P)/P$ and direct Huffman code rate $R_H(\mathcal{G}_1) =$ L(3 – P) [2], The reduced source $\mathcal{G}_1'$ is also geometrically distributed and has direct Huffman code rate $R_H(\mathcal{G}_1') = L(1-P)$. Substituting into (1), wc find that the ARH code rate for this source is

$$R_{ARH}(\mathcal{G}_1) = (1-P)[L(P) + L(1-P)].$$

This quantity approaches ∞ as P approaches zero. Thus the rate of ARH coding may be unbounded even though the redundancy is not.

The redundancy of direct Huffman coding on $\mathcal{G}_1$ is

$$\rho_H(\mathcal{G}_1) = L(1-P) - \frac{\mathcal{H}_2(P)}{P}.$$

and the redundancy of ARH is

$$\rho_{ARH}(\mathcal{G}_1) = (1-P)\,[L(P) + L(1-P)] - \frac{\mathcal{H}_2(P)}{P}.$$

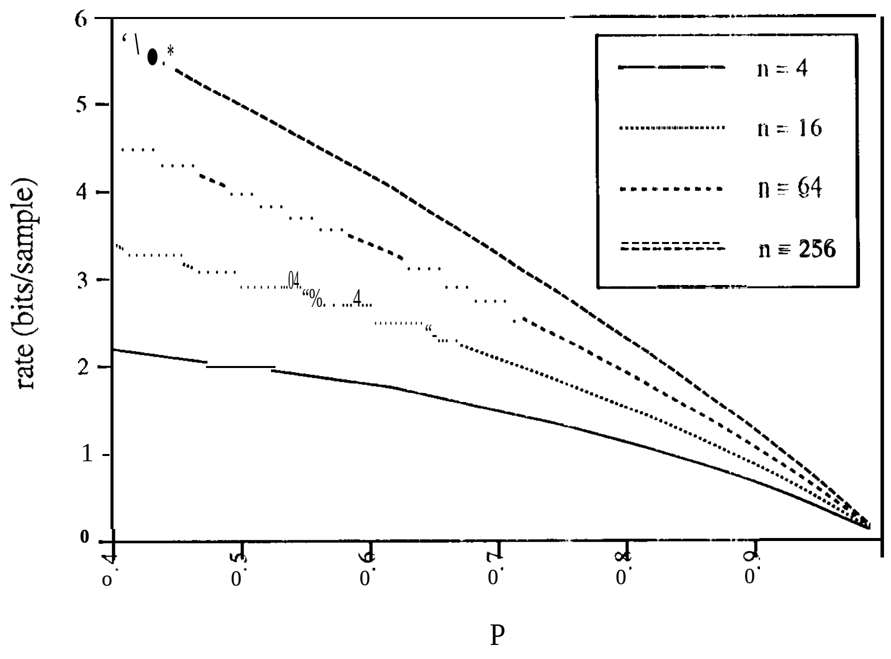

Figure 2: Upper bounds on ARH rate for alphabet size n.

These redundancies are shown in Figure 3. We find that for this source, the points where runlength Huffman coding becomes equal to and superior to direct Huffman coding are $P = 1 - \gamma$ and $P = \gamma$.

3.3 Two-sided Geometric Source

Consider the two-sided geometric source $\mathcal{G}_2$ with probability distribution

$$\mathrm{Prob}[i] = \begin{cases} P, & i = 0 \\ \left(\frac{1+P}{2}\right)\left(\frac{1-P}{2}\right)^{|i|}, & i \neq 0 \end{cases}$$

This is the distribution produced by a uniformly quantized Laplacian source with q/σ = --Win (1 – 1'), where q is the quantizer stepsize and σ^2 is the variance of the Laplacian distribution.

If $P \geq \sqrt{5} - 2 \approx .236$ then zero is the most probably source symbol. For this range of P, we plot in Figure 4 the redundancy for ARH and direct Huffman coding for a truncated version of $\mathcal{G}_2$ (eliminating the integers with magnitude exceeding 20). The graph indicates that for this source, direct Huffman coding is more efficient for $P < .394$, that ARH is more efficient when $P > \gamma$, and that the two schemes are equivalent when $.394 \leq P \leq \gamma$.

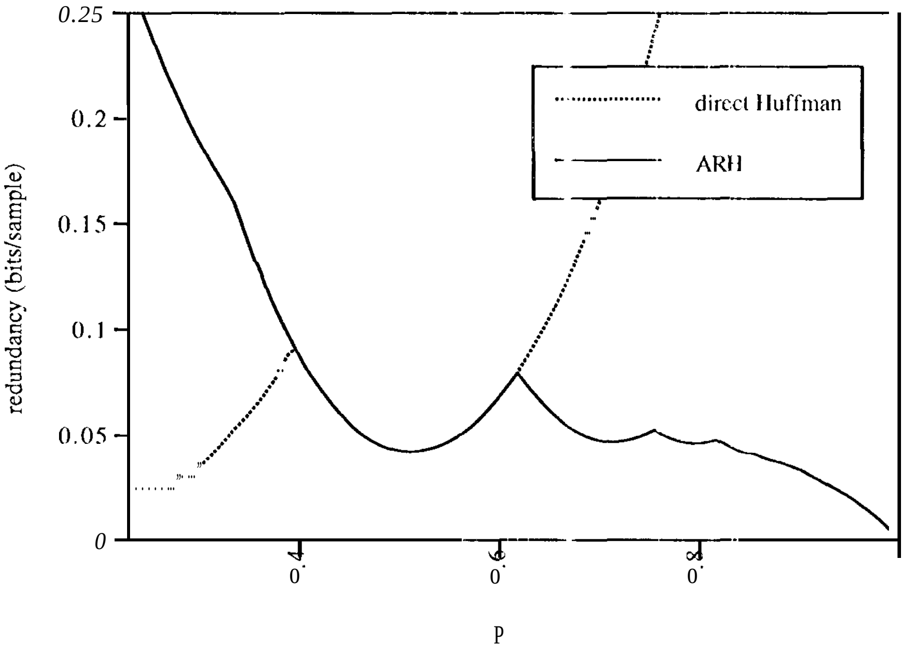

Figure 4: Redundancy of ARH and direct Huffman coding for two-sided Geometric source.

5. a block-orientation to alleviate memory requirement
6. efficiency over a wide range of entropy
7. it can be easily combined with other communications functionalities like error-containment and packetization

Acknowledgment

The authors are grateful to Padhraic Smyth for his helpful comments.

References

[1] S. Golomb, "Runlength Encodings," *IEEE Trans. Inform. Theory*, vol. 1'1'-12, pp.399-401, July 1996.

[2] R. Gallager and D. Van Voorhis, "optima] Source Codes for Geometrically Distributed Integer Alphabets," *IEEE Trans. Inform Theory*, vol. IT-21, pp. 228-230, March 1975.

[3] K. Cheung, "Efficiently Huffman Coding of a Discrete Independently and Identically Distributed Source with a Dominant Symbol - A Fast Implementation," *In preparation.*

[4] W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard,* New York, van Nostrand Reinhold, 1993.

[5] E. Adelson and E. Simoncelli, "Subband Image Coding with Three-tap Pyramids," Picture Coding Symposium, 1990, Cambridge, MA.

[6] S. Dolinar, K.-M. Cheung, I. Onyszchuk, F. Pollara and S. Arnold, "Compressed/Reconstructed Test Images for CRAF/Cassini," *The TDA Progress Report* vol. October) 42-104: September - December, 1990, Jet Propulsion Laboratory, Pasadena, February 15, 1991.

[7] "Variable-Length Coding of Quantized Block-Transformed Data," *in preparation.*

[8] T. Cover and J. Thomas, *Elements of Information Theory,* Wiley, 1991.

[9] O. Johnsen, "On the Redundancy of Binary Huffman Codes," *IEEE Tran. Inform. Theory,* vol. IT-26, pp. 220-222, March, 1980.

[10] R. Capocelli, R. Giancarlo, and I. Taneja, "Bounds on the redundancy of Huffman Codes," *IEEE Trans. Inform. Theory,* vol. IT-32, pp. 854-857, Nov. 1986.

[11] B. Montgomery and J. Abrahams, "On the Redundancy of Binary Prefix-Condition Codes for Finite and Infinite Sources," *IEEE Trans. Inform. Theory,* vol. IT-33, pp. 156-160, Jan. 1987.

[12] R. Capocelli and A. Santis, "Tight Upper Bounds on the Redundancy of Huffman Codes," *IEEE Trans. Inform. Theory,* vol. IT- 35, Sept. 1989.

[13] P. Smyth, "Entropy-Based Bounds on the Redundancy of Huffman Codes", *IEEE International Symposium on Information Theory,* San Diego, CA, Jan. 14-19, 1990.

[14] R. Yeung, "Local Redundancy and Progressive Bounds on the Expected Length of a Huffman Code," *IEEE Trans. Inform. Theory,* vol. 37, pp. 687-691, May 1991.

[15] R. Yeung, "Alphabetic Codes Revisited," *IEEE Trans. Inform. Theory, vol.* 37, pp 564-572, May 1991.

[16] D. Manstetten, "Tight Bounds on the Redundancy of Huffman Codes," *IEEE Trans. Inform. Theory,* vol. IT-38, no. 1, Jan. 1992, pp. 144-151.